Docket No.: 42390P11202
Express Mail No.: EL651821063US

UNITED STATES PATENT APPLICATION

FOR

METHOD AND APPARATUS FOR TRANSLATING GUEST PHYSICAL ADDRESSES IN A VIRTUAL MACHINE ENVIRONMENT

Inventors:

Andy Glew
Michael A. Kozuch
Erich S. Boleyn
Lawrence O. Smith III
Gilbert Neiger
Richard Uhlig

Prepared By:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Blvd., 7th Floor
Los Angeles, California 90025-1026
(310) 207-3800

METHOD AND APPARATUS FOR TRANSLATING GUEST PHYSICAL ADDRESSES IN A VIRTUAL MACHINE ENVIRONMENT

FIELD OF THE INVENTION

This invention relates generally to computers, and more particularly to computer memory address translation.

BACKGROUND

A computer processor accesses system memory to retrieve or store data within the system memory. Specifically, the processor uses the physical address of data in the memory to identify and access the data. However, the physical address where data is stored in the memory is not the address that the processor uses to index the data during internal manipulations. Rather, the processor assigns a virtual address to data being processed according to program instructions. Thus, memory accesses often require the translation of virtual addresses into physical addresses.

Many processors use virtual or demand-paged memory schemes, where sections of an execution environment of a program are mapped into physical memory as needed. Virtual memory schemes allow the use of physical memory much smaller in size than the virtual address space of the processor and also provide a mechanism for memory protection so that multiple programs sharing the same physical memory do not adversely interfere with each other.

In a virtual memory scheme, the virtual and physical address spaces are divided into blocks of contiguous addresses, so that virtual and physical addresses belong to

at most one block. The blocks can be of a constant size or can have variable sizes as dictated by system and/or program execution requirements.

These blocks are customarily referred to as pages if they are of a constant or fixed size. If variable sized blocks are used, the blocks are referred to as segments. Thus, the virtual address space may be divided into either segments or pages. A typical page size may be approximately 4 kilobytes.

DESCRIPTION OF THE DRAWINGS

Various embodiments are illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an," "one," or "various" embodiments in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

Figure 1 is a flow chart showing one method of translating a first address into a second address in a virtual machine environment.

Figure 2 is a logical diagram that illustrates how the matching and combining functions of various embodiments interact to accomplish virtual machine translation.

Figure 3 is a diagram of a computer system with a virtual machine translation ("VMTR") unit disposed in the central processing unit.

Figure 4 is a logical diagram that illustrates how the bitwise matching function of an embodiment is accomplished with a mask value and a base value.

Figure 5 is a logical diagram that illustrates how the combining function of an embodiment is accomplished with a mask value and an offset value.

Figures 6A and 6B illustrate the logic required by one embodiment to conduct virtual machine translation to obtain a translated address, a memory type for the translated address, and a fault bit for the translation.

Figure 7 is a logical diagram that illustrates an embodiment that verifies that each bit matches and that the appropriate memory type range register is active.

DETAILED DESCRIPTION

Various embodiments disclosed herein implement an efficient address translation scheme that yields a translated address, a memory type for the translated address, and a fault bit for the translation. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the various embodiments. It will be apparent, however, to one skilled in the art that the embodiments may be practiced without some of these specific details. For example, various signals, layout patterns and logical circuits may be modified according to the teachings of the various embodiments.

The following description and the accompanying drawings provide examples for the purposes of illustration. However, these examples should not be construed in a limiting sense as they are not intended to provide an exhaustive list of all possible implementations. In other instances, well-known structures and devices are omitted or simplified in order to avoid obscuring the details of the various embodiments.

Referring now to **Figure 1**, a flow chart is shown that illustrates a method of translating a first address into a second address in a virtual machine environment. In a

virtual machine environment, the physical addresses produced under the control of the operating system (referred to herein as the guest operating system) are referred to as guest physical addresses (e.g., first address in **Figure 1**). These guest physical addresses are translated into host physical addresses (e.g., second address in **Figure 1**) by a virtual machine monitor depending on the content of a memory type assigned to the memory range associated with the physical address.

The memory type can be stored in multiple data structures and accessed as necessary. Among other data, the memory type indicates whether any translation should occur for physical addresses from a given address range and whether accesses to a given address range should cause a fault. In various embodiments, an offset bit in the memory type indicates whether translation should occur. If no translation should occur, then the translated address is the same as the physical address. If translation is permitted to occur, the matching and combining functions described below are performed.

At block 10 of **Figure 1**, an interim first address is computed from a first address. The first address is associated with one of a plurality of segments of a physical address space (e.g., memory). Preferably, these segments are disjoint, meaning that the segments do not overlap.

Disjoint segments are desirable because each address only belongs to one segment. This eliminates the need for conflict resolution, which is necessary when two different mappings are produced by an address that belongs to two overlapping segments.

Assuming the memory type for the first address indicates that translation should occur, an interim base value is computed from a base value associated with the first address at block 12. The interim first address and the interim base value are compared at block 14. At decision block 16, it is determined whether the first address should be translated.

The decision at block 16 is based on whether the interim first address and the interim base value "match," as shown by the comparison carried out at block 14. If the interim first address and the interim base value match, the first address can be translated. The match function is performed to verify that a valid translation exists for the first address.

In various embodiments, the following equation is used to determine whether the interim first address and the interim base value match:

(Address&Mask)=(Base&Mask) Equation 1

The left side of the equation represents the interim first address, and the right side of the equation represents the interim base value. Specifically, a Mask value associated with the first address is separately applied (e.g., with a logical AND function) to both the first address and the base value.

Every bit position specified by Mask is retained for comparison. For example, if the Mask value is 11110000, then the four most significant bits of the first address (e.g., the interim first address once the Mask is applied) would be retained for comparison with the interim base value (e.g., the four most significant bits of the base value).

If the values on each side of the equation are equivalent, then the interim first address and the interim base value match. If not, there is not a valid translation for the first address.

It is worth noting that other matching mechanisms can be used. For example, a range check can be used to determine whether a first address can be validly translated.

Regardless of the matching mechanism used, if there is a match and the memory type indicates that an offset should be applied, the first address is "combined" with an offset value to obtain a second address (e.g., translated address) at block 20. If there is no match, a fault alert is issued

at block 18. The respective actions taken at block 20 and block 18 will be discussed in turn.

If a match occurs, the combining function of block 20 in one embodiment is a bitwise logical operation similar to the matching equation (e.g., Equation 1) described above. Specifically, the following equation is used to translate the first address, $Address_1$, to obtain the second address, $Address_2$:

$$Address_2=(Address_1\&\sim Mask)\,|\,(Offset\&Mask) \quad \text{Equation 2}$$

The "&" represents the logical AND function, and the "|" represents the logical OR function. The "~" represents the inverse of the value immediately following.

Thus, the Offset bits specified by Mask will replace the $Address_1$ bits specified to be replaced by ~Mask. For example, if $Address_1$ is AAAAAAAA, Offset is TTTTTTTT, and Mask is 11110000, then $Address_2$ will be TTTTAAAA. This result is due to the fact that Mask specified that the four most significant bits of Offset should replace the four most significant bits of $Address_1$, and ~Mask specified that the four least significant bits of $Address_1$ should be retained.

In other embodiments, the combining function of block 20 is accomplished by adding the offset value to the first address to obtain the second address.

Various embodiments further include determination of the memory type of the translated address. This determination process is conducted either concurrently with translation or after the translation is complete. In one embodiment, the memory type for the translated address is associated with the base value that matched the physical address.

Focusing now on block 18, a fault alert is issued to indicate that no mapping exists for the physical address. Typically, no mapping will exist for a physical address if the guest operating system attempts to access a region of memory that does not have a device or address that should respond to such an access attempt.

The fault alert can also entail a notification that an attempt has been made to access a particular segment (e.g., the segment to which the physical address belongs). For example, such a notification can be issued upon the detection of whether a fault bit has been set for the particular segment. Depending on the embodiment, the fault bit can be set in one or more values. In one embodiment, the fault bit is set in the memory type associated with the base value that matched the physical address to be translated.

Figure 2 is a logical diagram that shows one embodiment of the interaction between the matching and combining

functions described above. Although a 32-bit address is shown, the various embodiments described herein can be adapted for more than or less than 32 bits. Each bit of the 32 bit physical address, PA, is analyzed to determine whether there is a match. As long as all of the bits match, a translated address, TA, will be produced. However, if there is not a complete match, a fault notification may be issued depending on the system configuration.

For example, when a physical address bit matches, the matched bit is combined with the appropriate bit from the offset value, as previously described, to obtain a translated bit. The translated bit is combined with all of the other properly matched and translated bits to form the translated address.

Figure 3 shows an example of computer system 22 with virtual machine translation ("VMTR") unit 48 to perform address translation described above. Specifically, computer system 22 includes central processing unit ("CPU") 40, memory 42 coupled to central processing unit 40, virtual machine translation unit 48 disposed within central processing unit 40, and chipset 44. Although **Figure 3** shows virtual machine translation unit 48 disposed within central processing unit 40, it is contemplated to have virtual machine translation unit 48 located elsewhere within computer system 22 or even remote from, yet coupled to,

computer system 22. Computer system 22 communicates with external devices (e.g., keyboard, mouse, monitor, etc.) via input/output bus 46.

In an embodiment, computer system 22 includes memory 42, at least a portion of which is divided into a plurality of segments, comparison logic circuitry coupled to the memory, and combination logic circuitry coupled to the comparison logic circuitry and to the memory. Although not shown in **Figure 3**, the comparison logic circuitry and the combination logic circuitry are disposed within virtual machine translation unit 48.

Figures 4 and 7 show a logical representation of the comparison logic circuitry. Specifically, the comparison logic circuitry is designed to conduct the matching operation set forth in Equation 1 above. As shown in **Figure 4**, the interim first address is computed with AND gate 24, and the interim base value is computed with AND gate 26. Gate 28 is an inverted exclusive OR gate, which means that gate 28 determines whether the output from gate 24 and the output from gate 26 are equal. If so, there is a match. If not, there is no match.

This matching operation is carried out on a bit-by-bit basis and may be carried out in parallel. Once the matching operation is carried out for every bit (e.g., BitMatch[i] determined for all values of i), an overall match, $Match_N$,

is determined, as shown in **Figure 7**. Specifically, AND gate 64 verifies that every bit of the physical address matches. If every bit matches and the appropriate memory type range register, discussed in detail below, is active (e.g., identified by $Active_N$ signal), AND gate 66 generates output $Match_N$ to indicate that translation can occur. Alternatively, $Match_N$ can indicate that no translation should occur such that the final physical address is the same as the input physical address. This can happen as a result of the physical address not matching or the memory type range register being inactive.

Figure 5 shows a logical representation of the combination logic circuitry. Specifically, the combination logic circuitry is designed to conduct the combining operation set forth in Equation 2 above. AND gate 30 is used to determine which bits of the physical address, PA, are to be retained in the translated address, TA. AND gate 32 is used to determine which bits of the Offset value are to replace the physical address bits that are not to be retained in the translated address.

The translated address is calculated by OR gate 34. Pass gate 36 will only allow the signal from OR gate 34 to pass as the translated address if $Match_N$ (from **Figure 7**) indicates that the physical address should be translated. As indicated in **Figure 2**, the match function (**Figures 4 and**

7) and the combining function (**Figure 5**) are conducted on a bit-by-bit basis in various embodiments.

Figures 6A and **6B** show a logical diagram of an embodiment for Intel 32-bit architecture processors in which a translated address and memory type ("MT") are derived from the input physical address. Specifically, translation unit 50 includes table 52 of fixed-range memory type range registers ("MTRRs"). In the embodiment shown, these fixed range MTRRs provide memory types for addresses in the range between zero and one megabyte. However, this fixed range can vary. Here, zero to one megabyte is chosen because, historically, this range of memory has been highly fragmented.

Translation unit 50 also includes table 54 of variable range MTRRs. The variable range MTRRs define the memory type for a number of variable size ranges. The ranges are defined by a base value and a mask value. Each range is also associated with a valid bit, a memory type (which can include a fault bit and an offset bit, the offset bit to indicate whether or not to translate), and an offset value. These values are used, as described above, to determine whether to translate a first address and, if necessary, to translate the first address.

Fault detection logic 56 represents fault detection circuitry coupled to comparison logic circuitry (the logical

representation of which is shown in **Figures 4 and 7**). Fault detection logic 56 is configured to detect and issue fault alerts based on user preferences. As described above, the fault alerts can be issued if no mapping exists or if an attempt is made to access a particular segment.

MTRRdefType register 58 controls the operation of translation unit 50. Specifically, MTRRdefType register 58 has an enable field ("E") that controls whether the MTRRs, both fixed and variable, are active. The fixed enable field ("FE") of MTRRdefType register 58 controls whether the fixed range registers are enabled. Thus, when the enable field indicates that the MTRRs are active, translation can occur. If the MTRRs are inactive, the memory type specified by the Type field from MTRRdefType register 58 is used.

Thus, translation unit 50 takes an Input Physical Address (e.g., first address) and translates the address, if necessary, and outputs the Final Physical Address (e.g., second address), the memory type for the Final Physical Address, and a Fault Bit. Multiplexer 60 outputs a memory type from either table 52 of fixed range MTRRs, table 54 of variable range MTRRs, or from MTRRdefType register 58.

Multiplexer 62 outputs an address based on whether the offset bit is set. If the offset bit indicates that no translation should occur, the Final Physical Address will be the same as the Input Physical Address. If the offset bit

indicates that the address should be translated, the translated address obtained from table 54 of variable range MTRRs (e.g., via matching and combining described above) will be the Final Physical Address.

It is to be understood that even though numerous characteristics and advantages of various embodiments have been set forth in the foregoing description, together with details of structure and function, this disclosure is illustrative only. Changes may be made in detail, especially matters of structure and management of parts, without departing from the scope of the various embodiments as expressed by the broad general meaning of the terms of the appended claims.